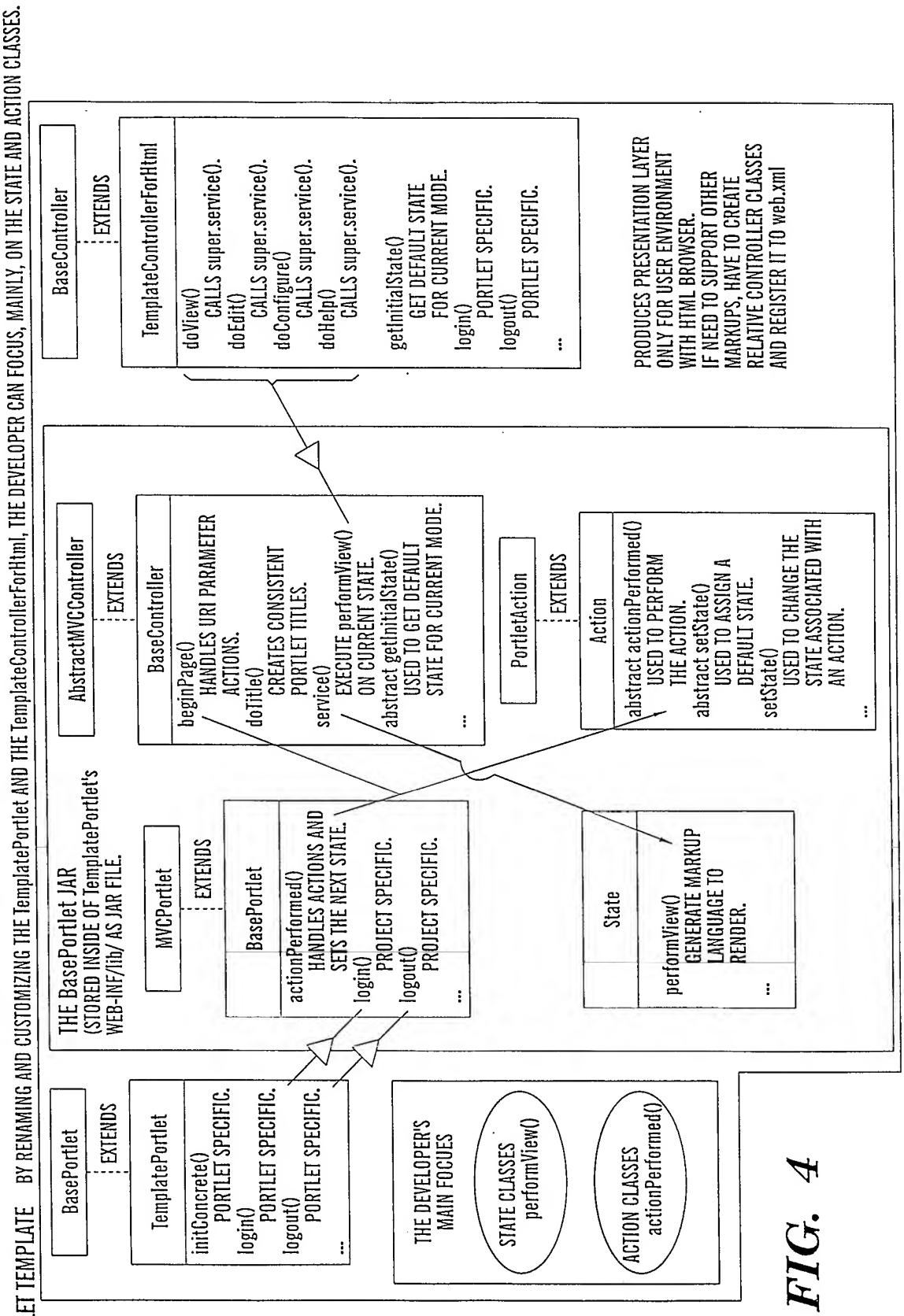
PORTLET TEMPLATE BY RENAMING AND CUSTOMIZING THE TemplatePortlet AND THE TemplateControllerForHtml, THE DEVELOPER CAN FOCUS, MAINLY, ON THE STATE AND ACTION CLASSES.
BasePortlet
EXTENDS
TemplatePortlet
initConcrete()
PORTLET SPECIFIC.
login()
PORTLET SPECIFIC.
logout()
PORTLET SPECIFIC.
...
THE DEVELOPER'S MAIN FOCUES
STATE CLASSES performView()
ACTION CLASSES actionPerformed()
THE BasePortlet JAR
(STORED INSIDE OF TemplatePortlet's WEB-INF/lib/ AS JAR FILE.
MVCPortlet
EXTENDS
BasePortlet
actionPerformed()
HANDLES ACTIONS AND SETS THE NEXT STATE.
login()
PROJECT SPECIFIC.
logout()
PROJECT SPECIFIC.
...
AbstractMVCController
EXTENDS
BaseController
beginPage()
HANDLES URI PARAMETER ACTIONS.
doTitle()
CREATES CONSISTENT PORTLET TITLES.
service()
EXECUTE performView() ON CURRENT STATE.
abstract getInitialState()
USED TO GET DEFAULT STATE FOR CURRENT MODE.
...
PortletAction
EXTENDS
Action
abstract actionPerformed()
USED TO PERFORM THE ACTION.
abstract setState()
USED TO ASSIGN A DEFAULT STATE.
setState()
USED TO CHANGE THE STATE ASSOCIATED WITH AN ACTION.
...
State
performView()
GENERATE MARKUP LANGUAGE TO RENDER.
...
BaseController
EXTENDS
TemplateControllerForHtml
doView()
CALLS super.service().
doEdit()
CALLS super.service().
doConfigure()
CALLS super.service().
doHelp()
CALLS super.service().
getInitialState()
GET DEFAULT STATE FOR CURRENT MODE.
login()
PORTLET SPECIFIC.
logout()
PORTLET SPECIFIC.
...
PRODUCES PRESENTATION LAYER ONLY FOR USER ENVIRONMENT WITH HTML BROWSER.
IF NEED TO SUPPORT OTHER MARKUPS, HAVE TO CREATE RELATIVE CONTROLLER CLASSES AND REGISTER IT TO web.xml

FIG. 4

Template Controller and Base Controller Source (Part A)

```
public class BaseController extends AbstractMVCController {
.
/*****************************************************
*   Public Method Name: service
*   Purpose:
*      This method is the method which is called when
*      the user selects to view the portlet. It is
*      responsible for rendering appropriately.
*
*    @param PortletRequest
*         Portlet request.
*    @param PortletResponse
*         Portlet response from view.
*    @return none
*    @throws PortletException
*    @throws IOException
*****************************************************/
public void service(PortletRequest request,
PortletResponse response)
  throws PortletException, IOException {

  try {
    //  Handle deferred exceptions
    processActionException(request);

    //  Get the portlet state object from session, if
    //  not there get the initial state for the mode.
    PortletSession session = request.getPortletSession();
    State nextState =
       (State) session.getAttribute(
          request.getMode().toString() +
          State.STATE_OF_WORKFLOW);
    if (nextState == null) {
       nextState = getInitialState(request.getMode());
       request.getPortletSession().setAttribute(
          request.getMode().toString() +
          State.STATE_OF_WORKFLOW, nextState);
    }
    //  Dispatch to state
    nextState.performView(
          baseClassService,
          request,
          response,
          getPortletConfig());
  } catch (Exception e) {

     //  Display any generated exceptions
     displayException(request, response, e);
```

FIG. 6A

Template Controller and Base Controller Source (Part B)

```
    }
  }
  }  //end Class

  public class TemplateControllerForHtml extends
  BaseController
  .
  {/************************************************
  *   Purpose:
  *         Below API method are handled as states.
  *
  *    @param PortletRequest
  *    @param PortletResponse
  *
  *    @return none
  *
  *    @throws PortletException
  *    @throws IOException
  ************************************************/
  public void doConfigure(PortletRequest request,
  PortletResponse response)
    throws PortletException, IOException {
     service(request, response);
  }

  public void doEdit(PortletRequest request, PortletResponse
  response)
     throws PortletException, IOException {
     service(request, response);
  }

  public void doHelp(PortletRequest request, PortletResponse
  response)
     throws PortletException, IOException {
     service(request, response);
  }

  public void doView(PortletRequest request, PortletResponse
  response)
     throws PortletException, IOException {
     service(request, response);
  }
  .
  }  //end Class
```

FIG. 6B